ROBOT TASK SPECIFICATION AND EXECUTION THROUGH RELATIONAL POSITIONING

Adolfo Rodríguez * **Luis Basañez** * **Enric Celaya** **

* *Institute of Industrial and Control Engineering (IOC), 08028 Barcelona, Spain*
** *Industrial Robotics Institute (IRI), 08028 Barcelona, Spain*

Abstract: This paper presents a relational positioning methodology for flexibly and intuitively specifying offline programmed robot tasks, and for assisting the execution of teleoperated tasks featuring precise or repetitive movements.
By formulating an object positioning problem in terms of symbolic geometric constraints, the movements of an object can be restricted totally or partially, independently of its initial configuration. To solve the problem, a 3D sequential geometric constraint solver called PMF –Positioning Mobile with respect to Fixed– has been developed. PMF exploits the fact that in geometric constraint sets the rotational component can often be decoupled from the translational one and solved independently.

Keywords: Robotics, Constraint Satisfaction Problems, Geometry, Robot Programming, Teleoperation.

1. INTRODUCTION

When asked to perform a positioning task, humans usually think of it in terms of satisfying geometric relations. For example, placing a glass on top of a table can be accomplished by making the bottom surface of the glass coincide with the tabletop. Positioning tasks can restrict not only totally, but also partially the movement of an object, so in the previous example, the glass can freely translate along directions parallel to the tabletop and can freely rotate about an axis perpendicular to it and still comply with the imposed relation. Furthermore, by using geometric relations, a task can be defined independently of the initial configurations of the involved objects, so if by some reason the initial positions of the glass and the table change, the task definition remains meaningful and need not be restated.

In the same way, many robot tasks require the positioning of objects with respect to their surroundings. Although this is a ubiquitous problem in robotics, most existing approaches fail to fulfill all the end user's needs, they are not intuitive enough, and rarely support the notions of partial movement restriction and initial configuration independence. For example, in traditional offline programming, configurations are defined in terms of non-intuitive parameters such as homogeneous transformations and joint space coordinates. In gestural programming, the burden is placed on an operator that manually moves the robot end-effector along the desired trajectories, trading a simpler interface with the implied downtime of the robot workcell and precision issues inherent to humans. Simulation-based programming is an attractive alternative since it imposes no workcell downtime, but its usefulness depends on a problem representation interface that is both intuitive and adapted to the end user's needs. Relational positioning can be used to define such an interface.

[1] This work was partially supported by the CICYT project DPI2005-00112.

Relational positioning is a powerful means for positioning objects in space, in which the problem is formulated in terms of geometric constraint sets. A geometric constraint is a relation (distance, angle, tangency, ...) between two or more geometric elements (points, curves, surfaces) that must be satisfied. These elements usually represent boundary or reference features of parent objects. For example, a point may represent the vertex of a cube, and a line may represent the axis of a cylinder. A geometric constraint solver is used to find the positions that each object should have in order to comply with these constraints. Relational positioning problems which can be solved by positioning one object at a time are called sequential.

Specifying a robot task can be done at multiple levels. Lower levels require defining all the details needed to complete the task (points, trajectories, and the like), while higher levels involve more abstract instructions leaving the details to automated processes. Relational positioning can be used to simplify task specification at low levels by using geometric constraints to define trajectory points, and at high levels by using the constraints as an intermediate layer between an automatic task planner and the robot controller. Mosemann and Wahl (2001), and Thomas and Wahl (2001) present a high level system for automatic offline planning and execution of assembly sequences that implements notions of relational positioning.

On the other hand, teleoperated task execution relies on operator skills. Some tasks involve movements that require precision or repeatability such as following a line or maintaining a fixed orientation. Turro *et al.* (2001) present a system that can generate forces on a haptic device to restrict its movement to curves and surfaces, but these must be explicitly defined by the operator. DeJong *et al.* (2006) use a combination of a structured light sensor system and an augmented-reality user interface to select curves and surfaces, which are then introduced to a constrained dynamic system simulation (Faulring *et al.*, 2005) for haptic rendering. In such situations geometric constraints required for the correct execution of the task can be defined, and their effect can be fed back to the operator via visual and haptic devices.

There exist many methods for solving geometric constraint problems (Hoffmann and Joan-Arinyo, 2005), most of which can be classified as graph-based, logic-based, algebraic –symbolic and numeric–, or hybrid. A method is said to be *complete* in a certain domain if it solves all solvable problems and detects unsolvable ones within the domain, and *general* if it can deal with any geometric constraint problem.

The application area for geometric constraint solvers is currently dominated by the CAD community, which has widely adopted them as an intuitive framework for parts and assembly design Most CAD solvers deal with 2D sketching problems (Fudos and Hoffmann, 1997; Owen, 1991), but there exist methods that model parts directly in 3D (Bruderlin, 1993; Kumar and Yu, 2001). Among other applications not so widespread figure mechanism design, kinematic modelling, molecular modelling, and robot task specification.

Kramer (1992) proposes a geometric constraint solver for open spatial kinematic chains and certain families of closed ones, but is not complete, specially in 3D problems. Porta *et al.* (2005) describe a complete and general numeric algebraic method based on Cayley-Menger determinants and branch-and-prune techniques. This method has the shortcoming that it is not well suited for relational positioning, because the problem cannot be directly formulated in terms of intuitive geometric constraints. Moreover, the form in which solutions are given needs to be processed before being used in a teleoperation or robot programming application.

This paper presents PMF –Positioning Mobile with respect to Fixed– a sequential geometric constraint solver for the relational positioning of rigid objects in 3D environments by means of distance and angular constraints between points, lines and planes; PMF handles under-, well-, and overconstrained (redundant and incompatible) problems. The solver has been integrated in a framework for specifying offline programmed robot tasks and assisting the execution of teleoperated ones.

PMF is based on the LMF solver (Celaya, 1992), and represents an optimization of this solver and an enhancement of its solving capabilities. The main improvements include: the ability to handle underconstrained problems; a broader set of input constraints including second-order distance constraints such as *point-point* and *point-line* distances; and an optimized constraint combination phase. PMF can be classified as *logic-based*, since it contains a set of constraint rewriting rules that transform the input constraints into a fundamental equivalent with known solution.

Two paramount requirements in the design of the solver have been the ability to handle underconstrained problems, since it is often desirable to restrict only partially the motion of a robot and to guide it using the available DOFs; and that solution computation should be fast enough to be included in high-frequency control loops and updated when the topology of the problem changes (e.g., moving obstacles).

Since there is a compromise between *completeness / generality* and *computational efficiency*, it has been opted for a solver that handles most practical problems of the application domain while fulfilling the above objectives and requirements. Such problems often turn out to be those whose solutions can be pictured qualitatively -but not quantitatively- by the user, so in most cases the user is able to naturally formulate the problem in a way that can be solved by the system.

A description of PMF and its features is presented in Section 2; sample problems are listed in Section 3; performance and implementation issues are covered in Section 4; and conclusions and future work are finally presented in Section 5.

2. SOLVER DESCRIPTION

The problem addressed by PMF is that of finding all possible configurations of a mobile object that satisfy a set of geometric constraints defined between the elements of the object and those of its (fixed) environment. The objects are defined in 3D space, they are assumed to be rigid and their positions known.

Many geometric constraints restrict both rotational and translational DOFs, but often they can be decomposed into pure rotational and pure translational constraints without losing its original meaning. An important observation is that the simultaneous satisfaction of two or more pure translational constraints may give rise to a rotational constraint. For example, a *point-point* coincidence is a pure translational constraint because it does not impose any rotational constraints by itself. However, the simultaneous satisfaction of two *point-point* coincidences between equally distant points implies the coincidence of the line-segments defined by them, which clearly imposes a rotational constraint.

The core idea behind the solver consists in obtaining all the implicit rotational constraints and using them to solve the rotational part first. Once the rotational solution subspace is obtained, the translation corresponding to each allowed orientation can be easily found. This approach may fail for problems that are not rotationally decoupled, that is, when it is not possible to obtain a set of pure rotational constraints. However, this is not the usual case in relational positioning and, when it appears, it is often easy for the user to provide additional constraints to make the problem solvable.

The solution process starts with the specification of the input constraints by the user. The input constraints have been selected with the aim of providing an easy way to define the problem. By means of coincidence/contained ($\in$ / $\ni$), distance (d), parallelism ($\|$), perpendicularity ($\perp$), and angle ($\angle$) relations between points, lines and planes, a set of thirty different input constraints can be constructed. Once the input constraints are specified, the following steps are performed: Constraint decomposition, constraint combination, and solution synthesis, each of which is explained next.

2.1 Constraint decomposition

Through constraint decomposition, input constraints are transformed into an equivalent set of fundamental constraints which are fewer and easier to work with. Fundamental constraints relate two geometric elements with a real-valued parameter, and can be either translational (distance) or rotational (angle). They follow the notation

$$\textbf{\textit{type}}(mobile\_el,\ fixed\_el,\ parameter),$$

where ***type*** becomes ***d*** for translational constraints or ***a*** for rotational constraints. There are five translational and one rotational fundamental constraints:

$\boldsymbol{d}(\mathbf{P}_m, \mathbf{P}_f, p)$: *point-point* distance,
$\boldsymbol{d}(\mathbf{P}_m, \mathcal{L}_f, p)$: *point-line* distance,
$\boldsymbol{d}(\mathbf{P}_m, \Pi_f, 0)$: *point-plane* coincidence,
$\boldsymbol{d}(\mathcal{L}_m, \mathbf{P}_f, p)$: *line-point* distance,
$\boldsymbol{d}(\Pi_m, \mathbf{P}_f, 0)$: *plane-point* coincidence,
$\boldsymbol{a}(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f, \alpha)$: *vector-vector* angle,

When the distance value on *point-point*, *point-line*, and *line-point* distance constraints equals zero (coincidence), they become more restrictive. The same happens with *vector-vector* angle constraints, yielding the case of parallel vectors.

As an example, the *line-line* coincidence input constraint $d(\mathcal{L}_m, \mathcal{L}_f) = 0$ is replaced by a pure translational *point-line* coincidence constraint $\boldsymbol{d}(\mathbf{P}_m, \mathcal{L}_f, 0)$ –where $\mathbf{P}_m \in \mathcal{L}_m$–, and a pure rotational *vector-vector* parallelism constraint between the line direction vectors $\boldsymbol{a}(\hat{\mathbf{d}}_{\mathcal{L}_m}, \hat{\mathbf{d}}_{\mathcal{L}_f}, 0)$.

2.2 Constraint combination

Constraints are tested in pairs. Translational constraints are combined first in order to complete the set of rotational constraints. If two translational constraints implicitly define rotational constraints, these are made explicit; if two constraints can be substituted by a more specific one, this new constraint must also pass through a combination round; if a redundant constraint is found, it is simply removed; if an incompatibility is found, the current constraint set is labeled as ill-defined and will have no solution.

As an example, consider two *point-plane* coincidence constraints $\boldsymbol{d}(\mathbf{P}_m, \Sigma_f, 0)$, $\boldsymbol{d}(\mathbf{Q}_m, \Pi_f, 0)$; and let $d_m = d(\mathbf{P}_m, \mathbf{Q}_m)$ and $d_f = d(\Sigma_f, \Pi_f)$.

When $\Sigma_f \parallel \Pi_f$ the following cases arise:

- If $d_m < d_f$ the constraints define an incompatible configuration (figure 1a).
- If $d_m \geq d_f > 0$, the second constraint can be substituted by the implicit rotation $\boldsymbol{a}(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f, \alpha)$, where $\hat{\mathbf{u}}_m$ points in the direction of $\overrightarrow{\mathbf{P}_m\mathbf{Q}_m}$, $\hat{\mathbf{u}}_f$ is the normal vector of Σ_f that points towards Π_f, and $\alpha = \cos^{-1}(d_f/d_m)$ (figure 1b).
- If $d_m > d_f = 0$, the second constraint can be substituted by $\boldsymbol{a}(\hat{\mathbf{u}}_m, \hat{\mathbf{n}}_{\Sigma_f}, \pi/2)$, where $\hat{\mathbf{n}}_{\Sigma_f}$ is a vector normal to Σ_f.
- If $d_m = d_f = 0$ the constraints are redundant (equal), so one is removed.

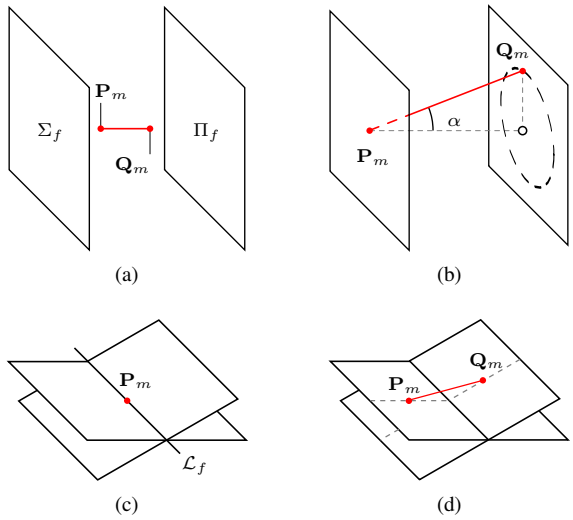


Fig. 1. Sample combinations of two *point-plane* coincidence constraints $\boldsymbol{d}(\mathbf{P}_m, \Sigma_f, 0), \boldsymbol{d}(\mathbf{Q}_m, \Pi_f, 0)$.

and when $\Sigma_f \nparallel \Pi_f \longrightarrow \Sigma_f \cap \Pi_f = \mathcal{L}_f$:

- If $d_m = 0$, both constraints can be substituted by the more restrictive $\boldsymbol{d}(\mathbf{P}_m, \mathcal{L}_f, 0)$ (figure 1c).
- If $d_m \neq 0$, no new constraints are obtained (figure 1d).

Once the combination of translational constraints is finished it is assumed that all rotational constraints have been made explicit. Rotational constraints can be now combined following guidelines similar to the ones used for translational constraints.

Sometimes the combination of two constraints may yield more than one valid alternative. When this occurs, the current fundamental constraint set is branched and each branch is treated as a separate problem. While one branch may prove to be incompatible, the remaining may still have valid solutions.

2.3 Solution synthesis

Once the fundamental constraint sets have been reduced to minimal expressions, solutions are independently computed for each set. Each constraint set can give rise to more than one solution. The computation is performed in two steps. First, the rotational component is solved, and if there exist rotational DOFs, a particular rotation is selected by initializing the free parameters. Then, the translational component is solved for the selected rotation, and, as above, if there exist translational DOFs, the free parameters are initialized. Notice that solutions are built on demand depending on the values of the free parameters, and that the translational component must be recomputed every time the rotations change.

Each solution defines a subspace of the six dimensional space of three translations and three rotations. The dimension of this subspace corresponds to the number of available DOFs. A selected instance of a given solution is represented by a rigid transformation parameterized by as many parameters as available DOFs. Additional information such as the current solution subspace can also be obtained. The rotational component of the subspace is characterized by unit vectors that represent rotation axis and a set of valid angular intervals, and the translational component of the subspace is characterized by curve or surface equations.

3. SAMPLE PROBLEMS

Two sample problems are presented in order to illustrate how PMF can be applied to the specification and execution of robot tasks. The first example shows how to specify a simple offline programmed assembly task using sets of geometric constraints, featuring how to define trajectory reference points with well-constrained configurations. The second example consists of a teleoperated painting task, and it shows how task execution can be improved by partially restricting the motion of the operator to a subspace of interest. In Section 4 it will be shown how the constraints can be interactively defined by means of an intuitive graphical interface.

3.1 Assembly task

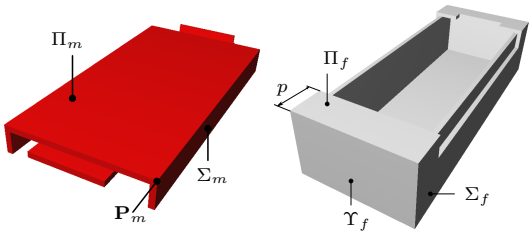
Consider the objects *cover* and *base*, depicted in figures. 2a and 2b, respectively. The task consists on securing *cover* (which is being grasped by a robot arm) to *base* as shown in figure 2c. Notice that since *cover* features a spring-activated locking mechanism, it must be positioned following the right sequence of configurations rather than directly heading to its final state. Consider now the following sequence of configurations for successfully performing the task:

(a) Position *cover* at an angle with respect to *base* ensuring the objects do not collide (figure 3a).
(b) Align *cover* with the rails on *base*'s sides (figure 3b).
(c) Translate *cover* along the rails (figure 3c).
(d) Rotate *cover* until the spring on its front locks (figures 3d and 2c).

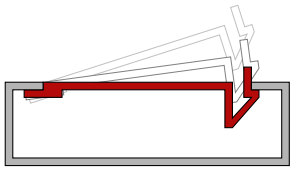
Configuration (a) (figure 3a) is represented with a constraint set that fully restricts *cover*'s motion:

$$\begin{aligned} \Sigma_m &= \Sigma_f \\ d(\mathbf{P}_m, \Pi_f) &= d_v \\ d(\mathbf{P}_m, \Upsilon_f) &= p + d_h \\ \angle(\Pi_m, \Pi_f) &= \alpha \end{aligned}$$

Configurations (b), (c), and (d) are then reached by sequentially setting to zero d_v, d_h, and α, respectively (figures 3b-3d).

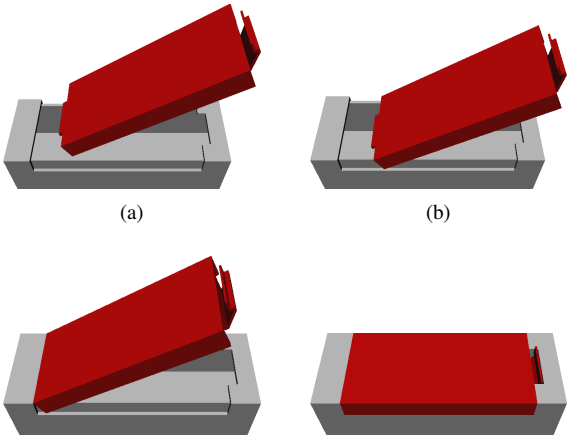


(a) Cover (mobile object) (b) Base (fixed object)



(c) Final configuration

Fig. 2. Assembly task consisting on positioning *cover* (a) with respect to *base* (b) as shown in (c).



(a) (b) (c) (d)

Fig. 3. Sequence followed to accomplish the assembly. Robot gripper is handling *cover*, but is not shown for clarity.

It must be stressed that the map between geometric constraint sets and solutions is *not* injective, so there exist other constraint sets that yield the same solutions as the above example.

One important remark is that since the constraints are defined symbolically, the task need not be redefined if the initial configurations of *cover* or *base* change, since the solver automatically recomputes the new solutions. This enables changing the distribution of the robot workcell with minimal downtime and no task reprogramming.

3.2 Teleoperated spray-painting task

Consider a spray-painting teleoperated task in which an operator must paint a free curve on a plane surface. Suppose now that to ensure optimal paint covering the nozzle of the spray painter must remain at a fixed constant distance from the painted surface as well as normal to it. If the operator is to perform successfully

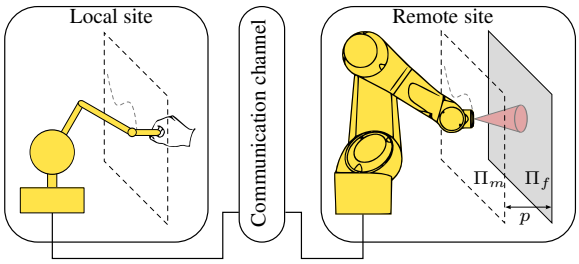


Fig. 4. A teleoperated spray-painting task.

the task, he/her must simultaneously trace the desired curve while satisfying the above relations.

Geometric constraints can be used to enforce the painting distance and orientation conditions, hence lowering the burden on the operator and permitting him/her to concentrate on tracing the curves. A scheme depicting the task is shown in figure 4.

By defining the constraint $d(\Pi_m, \Pi_f) = p$, the plane Π_m, that contains the nozzle and is normal to it, will remain parallel and at a distance p from the painting plane Π_f. The solution subspace, which permits translations along a plane and rotations about its normal, is sent to a haptic guidance module that generates virtual forces on the operator to maintain it inside the subspace. Details on how this module works as well as further examples like peg-in-hole insertions and following a guide can be found in (Nuño *et al.*, 2006; Nuño and Basañez, 2006).

4. PERFORMANCE AND IMPLEMENTATION ISSUES

The current implementation of the PMF solver is written in C++. Its performance was measured on a desktop PC with an Intel Pentium 4 processor running at 3.4GHz. The solution times range from 0.05ms for a simple problem featuring one constraint and one solution, to 1.5ms for a fully constrained problem featuring five constraints and thirty two distinct solutions. In contrast, methods featuring iterative processes are more sensitive to singularities and degeneracies, and tend to have much greater variations in their solution times. Comparatively, the examples in (Porta *et al.*, 2005) featuring one mobile object have solution time variations of up to five orders of magnitude, as opposed to two for PMF.

Figure 5 shows a screenshot of the solver user interface, featuring a 3D model of the task (right), and a control panel (left) that permits the graphical interactive definition of input constraints, displays information regarding the solutions of the current problem, and permits moving the mobile object along its current solution subspace by means of a set of knobs or a haptic device.

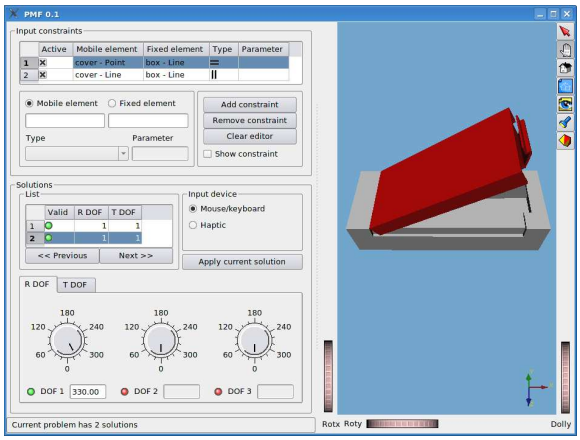


Fig. 5. Screenshot of the solver user interface.

5. CONCLUSIONS AND FUTURE WORK

A relational positioning methodology for flexibly and intuitively specifying offline programmed robot tasks, as well as for assisting the execution of teleoperated tasks that feature precise or repetitive movements has been presented.

It was shown how an object positioning problem can be formulated in terms of geometric constraints for restricting totally or partially the movements of an object independently of its initial configuration, and PMF, a 3D sequential geometric constraint solver was proposed as a means to find the solutions to such problem. The solver can handle under-, well-, and overconstrained problems with multiple solutions; and although it is not complete, the solvable subset handles most of the problems a user would be interested in. Experimental data shows that the solution times of the current implementation allow real-time interaction with a human operator and the inclusion of the solver in high-frequency loops that require response times within the millisecond order of magnitude.

Future lines of work comprise extending the solver's capabilities for handling multiple mobile objects and interfacing it with kinematics and path planning modules to enable the specification and execution of more complex (multi)robot tasks.

REFERENCES

Bruderlin, B. (1993). Using geometric rewrite rules for solving geometric problems symbolically. In: *Theoretical Computer Science 116*. Elsevier Science Publishers B.V.. pp. 291–303.

Celaya, E. (1992). Geometric Reasoning for the Determination of the Position of Objects Linked by Spatial Relationships. PhD thesis. Universitat Politècnica de Catalunya. Barcelona, Spain.

DeJong, B. P., E. L. Faulring, J. E. Colgate and M. A. Penshkin (2006). Lessons learned from a novel teleoperation testbed. *Industrial Robot: An International Journal* **33**(3), 187–193.

Faulring, E. L., K. M. Lynch, J.E. Colgate and M. A. Penshkin (2005). Haptic interaction with constrained dynamic systems. In: *IEEE Int. Conf. Robot. Automat.*. Barcelona, Spain. pp. 2458–2464.

Fudos, I. and C.M. Hoffmann (1997). A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics* **16**(2), 179–216.

Hoffmann, C.M. and R. Joan-Arinyo (2005). A brief on constraint solving. *CAD&A* **2**(5), 655–664.

Kramer, G. (1992). *Solving Geometric Constraint Systems*. MIT Press.

Kumar, A.V. and L. Yu (2001). Sequential constraint imposition for dimension-driven solid models. *Computer-Aided Design* **33**(6), 475–486.

Mosemann, H. and F.M. Wahl (2001). Automatic decomposition of planned assembly sequences into skill primitives. *IEEE Trans. Robot. Automat.* **17**(5), 709–718.

Nuño, E., A. Rodríguez and L. Basañez (2006). Force reflecting teleoperation via ipv6 protocol with geometric constraints haptic guidance. In: *Springer Trans. in Advanced Robotics – Advances in Telerobotics* (Springer Verlag, Ed.). Chap. 26. To be published.

Nuño, E. and L. Basañez (2006). Haptic guidance with force feedback to assist teleoperation systems via high speed networks. In: *37th International Symposium on Robotics* (International Federation of Robotics, Ed.). VDI Wissensforum IWB GmbH. Munich, Germany. pp. 1–14.

Owen, J.C. (1991). Algebraic solution for geometry from dimensional constraints. In: *Symposium on Solid Modeling Foundations and CAD/CAM Applications* (R. Rossignac and J. Turner, Eds.). ACM Press. Austin, TX. pp. 397–407.

Porta, J. M., L. Ros, F. Thomas and C. Torras (2005). A branch-and-prune solver for distance constraints. *IEEE Trans.Robot.* **21**(2), 176–187.

Thomas, U. and F.M. Wahl (2001). A system for automatic planning, evaluation and execution of assembly sequences for industrial robots. In: *IEEE Int. Conf. Intell. Robots Syst.*. Maui, Hawaii. pp. 1458–1464.

Turro, N., O. Khatib and E. Coste-Maniere (2001). Haptically augmented teleoperation. In: *IEEE Int. Conf. Robot. Automat.*. Seoul, Korea. pp. 386–392.